

IM3080 Design and Innovation Project (AY20xx/xx Semester x)

Individual Report

Name: Max Toh

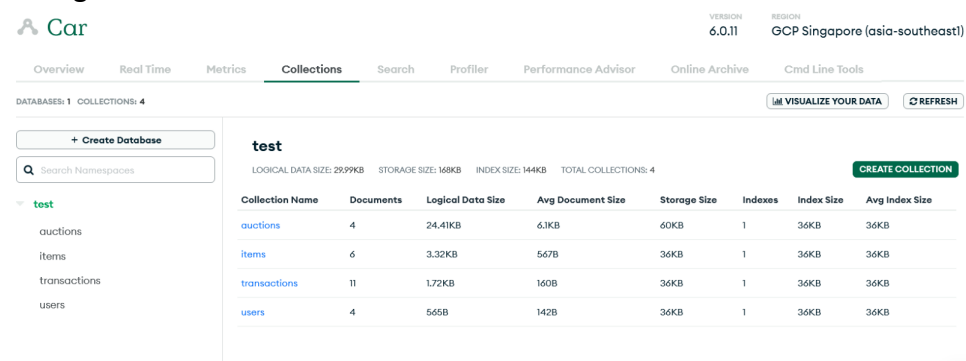
Group No: 5

Project Title: CLUTCH

Contributions to the Project (1 page)

In this project, I was part of the backend team and helped to create a database of the application by setting up the database on MongoDB, using it for the MERN stack. I also helped to develop the simple base APIs for the 4 tables in the database, the item list, the user list, the auction list and the transaction list. Also I helped to populate the database with entries so that the frontend can verify the data is working as intended.

MongoDB Database



The screenshot shows the MongoDB Cloud console interface. At the top, it displays the project name 'Car' and the version '6.0.11'. The 'Collections' tab is selected, showing a list of collections under the 'test' database. The collections are 'auctions', 'items', 'transactions', and 'users'. Each collection has a row of statistics: Documents, Logical Data Size, Avg Document Size, Storage Size, Indexes, Index Size, and Avg Index Size.

Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
auctions	4	24.41KB	6.1KB	60KB	1	36KB	36KB
items	6	3.32KB	567B	36KB	1	36KB	36KB
transactions	11	1.72KB	160B	36KB	1	36KB	36KB
users	4	565B	142B	36KB	1	36KB	36KB

Item Base API

```
const getAllCars = async (req, res) => {
  try {
    // res.json({msg: 'GET all cars'})
    const cars = await Items.find({}).sort({createdAt: -1});
    res.status(200).json(cars)
  } catch (error) {
    console.error(error);
    res.status(400).json({ error: 'Server error' });
  }
}
```

```
const getItem = async (req, res) => {
  const {seller} = req.params
  const itemList = await Items.find({seller: seller}).sort({createdAt:-1})

  res.status(200).json(itemList)
}
```

```
const getSoldItems = async (req, res) => {
  const {seller} = req.params
  const soldItems = await Items.find({seller: seller, sold: true}).sort({createdAt:-1})

  if (!soldItems) {
    return res.status(400).json({error: 'No previous items'})
  }

  res.status(200).json(soldItems)
}
```

```
const createItem = (async (req, res) =>{
  const {brand, model, colour, fuel_type, mileage, price, description, years_used, re

  try {
    const item_list = await Items.create({brand, model, colour, fuel_type, mileage,
    res.status(200).json(item_list)
    console.log(json(item_list))
  } catch (error) {
    res.status(400).json({error: error.message})
    console.log(error.message)
  }
})
```

```
const updateItem = async(req, res) =>{
  const {id} = req.params

  if (!mongoose.Types.ObjectId.isValid(id)) {
    return res.status(404).json({error: 'No such id'})
  }
  const item_list = await Items.findOneAndUpdate({_id: id},{
    ...req.body
  })
  if (!item_list) {
    return res.status(400).json({error: 'No previous items'})
  }
  res.status(200).json(item_list)
}
```

User Base API

```
const getUser = async(req, res) => {
  const {id} = req.params

  if (!mongoose.Types.ObjectId.isValid(id)) {
    return res.status(404).json({error: 'No User'})
  }

  const user = await User.findById(id)

  if(!user) {
    return res.status(404).json({error: 'No User'})
  }

  res.status(200).json(user)
}
```

```
const createUser = async (req, res) =>{
  const {name, contact_number, email} = req.body

  try {
    const user = await User.create({name, contact_number, email})
    res.status(200).json(user)
  } catch (error) {
    res.status(400).json({error: error.message})
    console.log(error.message)
  }
}
```

Auction Base API

```
const getAuction = async (req, res) =>{
  const {id} = req.params
  const auctionList = await Auction.find({_id : id}).sort({createdAt:-1})

  res.status(200).json(auctionList)
}

const createAuction = (async (req, res) =>{
  const {brand, model, colour, fuel_type, mileage, buyout_price, starting_bid, endi

  try {
    const auctionList = await Auction.create({brand, model, colour, fuel_type, mi
    res.status(200).json(auctionList)
    console.log(json(auctionList))
  } catch (error) {
    res.status(400).json({error: error.message})
    console.log(error.message)
  }
})
```

Transaction Base API

```
const getTransaction = async (req, res) =>{
  const {item_id} = req.params
  const transactionList = await Transaction.find({item_id : item_id}).sort({createdAt:-1})

  res.status(200).json(transactionList)
}

const createTransaction = (async (req, res) =>{
  const {item_id, user_id, bid_price} = req.body

  try {
    const transactionList = await Transaction.create({item_id, user_id, bid_price})
    res.status(200).json(transactionList)
    console.log(json(transactionList))
  } catch (error) {
    res.status(400).json({error: error.message})
    console.log(error.message)
  }
})
```

Reflection on Learning Outcome Attainment

Reflect on your experience during your project and the achievements you have relating to at least two of the points below:

- (a) Engineering knowledge
- (b) Problem Analysis
- (c) Investigation
- (d) Design/development of Solutions
- (e) Modern Tool Usage
- (f) The Engineer and Society
- (g) Environment and Sustainability
- (h) Ethics
- (i) Individual and Team Work
- (j) Communication
- (k) Project Management and Finance
- (l) Lifelong Learning

Point 1: (d) State the area: Design/Development of Solutions

In this project, There was a need to figure out how our frontend would interact with our backend, thus basic APIs were created to handle any possible request the user would have when interacting with the app, where basic function like creating new listings or updating current ones or seeing what is currently available is designed as simple as possible to ensure that the process can be verified to be working, so that later on, it can be made more complex to handle the specifications of the user. I learnt that it is better to build the functionalities as simple as possible to begin with and slowly add more features to suit the needs of the users.

Point 2: (e) State the area: Modern Tool Usage

In this project, I learnt about how to use both Github and MongoDB, I learnt about how to do pull and push requests in Github and how to set up a cloud database in MongoDB just to contain all the data needed for this application to function. I also learnt about the javascript in the form of NodeJS and ExpressJS to create the backend infrastructure for the application. Both the online applications and javascript can help me in the future, when another project might need storage in the form of source code in Github and data entries in MongoDB and how it can be interacted between the client and server through NodeJS and ExpressJS.